Accelerating Time-Varying Hardware Volume Rendering Using TSP Trees and Color-Based Error Metrics

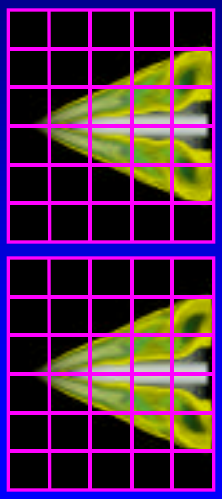
David Ellsworth Ling-Jen Chiang
AMTI/NASA Ames Research Center
Han-Wei Shen
The Ohio State University

Motivation

- Rendering large time-varying volumes is slow
- Can accelerate with hardware texturing
- Limited texture memory limits speed
  - separate texture memory
  - must load texture memory during rendering

Using Coherence to Accelerate Rendering

- Spatial coherence
  - region does not vary much spatially
  - flat shade region, or don't render if transparent
- Temporal coherence
  - region does not vary much over time
  - share region's texture between time steps



Approach

- Use Time-Space Partitioning (TSP) tree to identify coherence
  - by computing per-region error metrics
  - allows specification of allowable error
- Use error metrics based on voxel color instead of voxel value
- Modify texture hardware volume rendering algorithm
  - regular grids only

Related Work

- Volume rendering using texturing hardware

  Cullip and Neuman, 1993; Cabral, Cam, and Foran, 1994; SGI Volumizer 1998

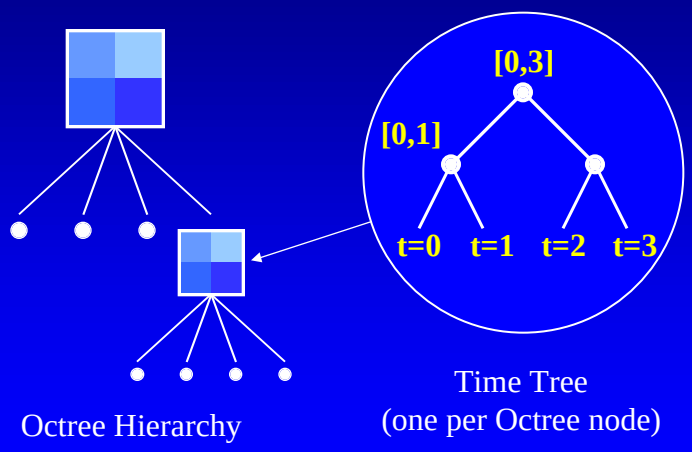
- Spatial hierarchies to find spatial coherence

  Levoy, 1990; Laur and Hanrahan, 1991; Wilhems and Van Gelder, 1994; Lamar *et al.*, 1999

- First TSP tree paper

  Shen, Chiang, and Ma, 1999

TSP Tree Data Structure



Octree Hierarchy

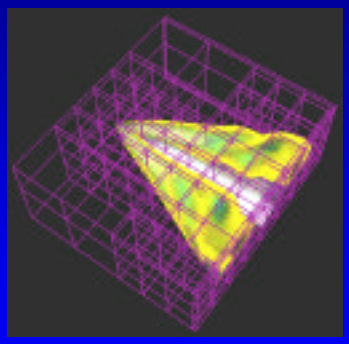
Time Tree
(one per Octree node)

TSP Tree Creation Algorithm

- Create octree skeleton and time trees
- Compute per-node temporal and spatial statistics and error metrics
- Update color-based error metrics when transfer function changes

TSP Tree Traversal Algorithm

- Compares error values in tree with error tolerances
- Finds set of nodes that covers volume and meets error tolerances
- Returns nodes largest in time and space



Minimum Subvolume Size

- Octree divides volume into subvolumes
- What size of subvolume?
  - smaller subvolumes find more coherence
  - larger subvolumes have less overhead
    - replicated voxels at boundaries
    - per-texture management
    - polygon generation and rendering
- We use 32x32x32 subvolumes

Error metrics

- Scalar metrics:
  - spatial: standard deviation of voxels over subvolume
  - temporal: average of per-voxel standard deviations over time
  - can be precomputed
- Reference color color:
  - modify standard deviation to use alpha-weighted distance between colors in RGB space
  - very slow: 4-20 minutes

Approximate Color Error Metric

- Does not compute voxel colors
- Spatial metric:
  - assumes scalar values are normally distributed
  - uses precomputed average and variance of scalars
  - recast standard deviation calculation to use distribution's population counts
  - fast: only iterates over transfer function entries

Approximate Color Error Metric

- Temporal metric is product of:
  - average difference between adjacent color table entries
  - average per-voxel standard deviation over time
- Fast: we compute in at most 0.3 seconds

Implementation

- Fast incremental polygon slicing algorithm (Yagel *et al.*, 1996)
- Creates textures for each time step, then reuses them (cached/non-cached)
  - via OpenGL glBindTextureExt

Experiments

- On Onyx^2 InfiniteReality^2, 64 MB texture memory, 1GB main memory
- Three error tolerances
  - zero
  - slight: has hard-to-notice artifacts
  - moderate: has unobjectionable artifacts
- Compared TSP & non-TSP algorithms using different error metrics
  - comparable error tolerances

Data Sets

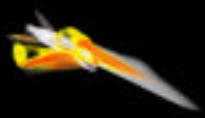
Sparse Delta

20 and 66 Mvoxels

Filled Delta

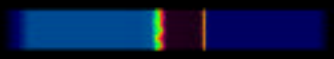
20 and 66 Mvoxels

F18



64 Mvoxels

Shock

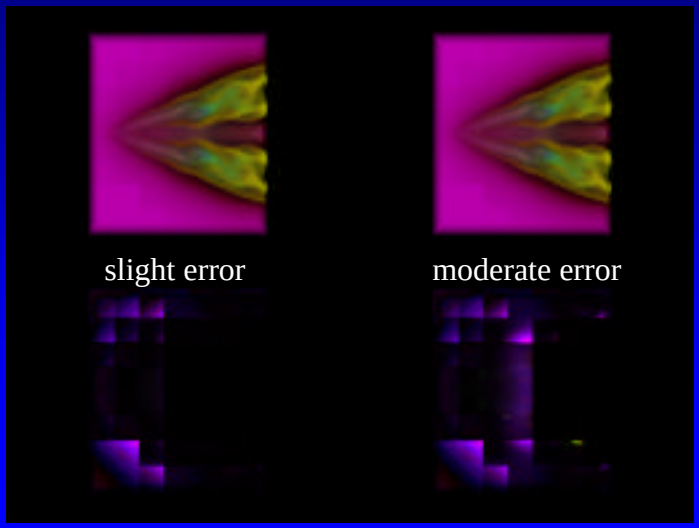


50 Mvoxels

Error Tolerance Example

Image with error allowed

Contrast-enhanced difference from zero error



slight error

moderate error

Video

- Typical user interaction
- Algorithm visualization

Error Tolerance Visualization

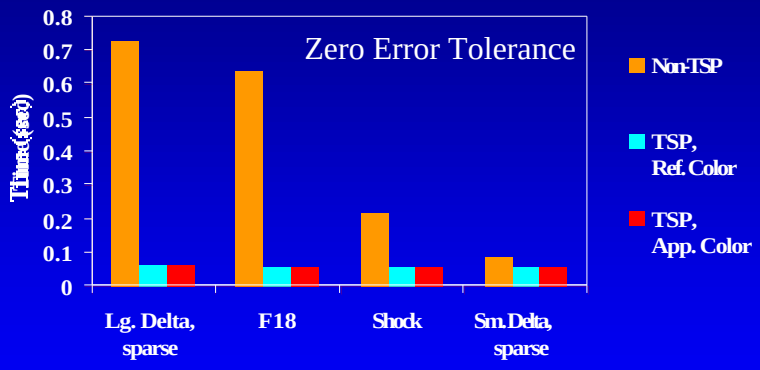
Results Summary

- TSP tree runs almost always faster when coherence exists
- Slower if no coherence due to overhead
- Non-zero error tolerance allows sharing of textures
- Reference and approximate color error metrics have very similar performance

Times While Creating Textures

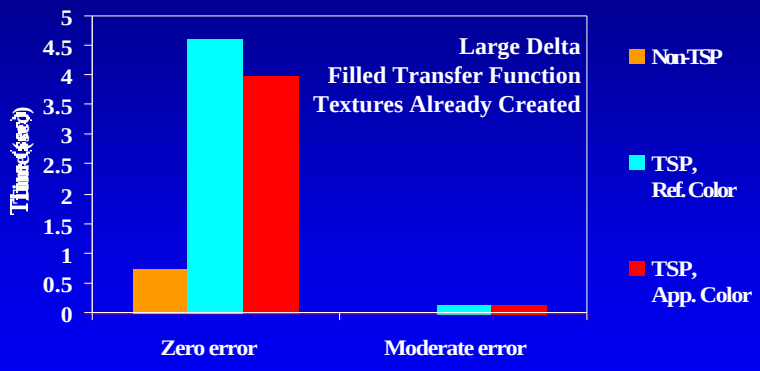
- Scalar error metric times range from 0.4 to 9 sec

Times When Textures Created



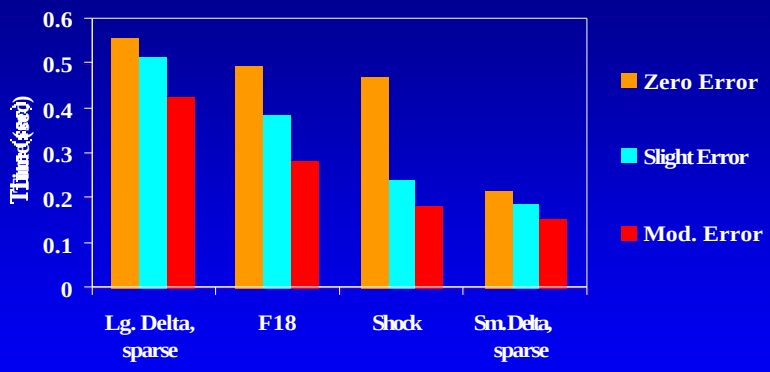
- Scalar error metric times range from 0.1 to 4.5 sec

Difficult Case



- Small texture savings with zero error tolerance
- Algorithm could fall back on non-TSP algorithm

Times with Error Allowed



- Approximate color error metric, textures created

Conclusions

- TSP tree algorithm accelerates hardware volume rendering
  - uses coherence to reduce texture usage and reduce rendering time
- Color-based error metrics
  - more effective at finding coherence than scalar-based metrics
  - approximate color error metric is effective and fast

Future Work

- Fall back to non-TSP algorithm when no coherence
- Error metrics using perceptual color space
- Simplification of TSP traversal algorithm

Acknowledgements

- Data sets
  - Neal Chaderjian
  - Ken Gee
  - Scott Murman
  - Ravi Samtaney
- Work supported by NASA Contracts NAS2-14303 and DTTS59-99-D-00437/A61812D